

# Tehnici Web

## CURSUL 3

Semestrul I, 2024-2025  
Carmen Chirita

# CSS Layout – proprietatea **position**

Specifica tipul de pozitionare al unui element în pagina;  
poate avea una din valorile:

- **position:static**
- **position:relative**
- **position:absolute**
- **position:fixed**
- **position:sticky**

Pentru pozitionare se folosesc proprietatile **left**, **right**, **top** și **bottom**  
Se va specifica mai intai tipul de pozitionare cu proprietatea **position**

<https://developer.mozilla.org/en-US/docs/Web/CSS/position>

# CSS Layout – proprietatea **position**

## position: static

Este poziția implicită a oricărui element HTML; cu poziția static elementul va avea un flux normal în pagină;  
Elementele cu poziția static nu sunt afectate de proprietățile left, right, top, bottom.

```
<style>
div{position:static;
    width:200px;
    border: 3px solid #73AD21;
    background-color: cyan;
}
</style>
</head>
<body>
<h2>Pozitionare statica</h2>
<p>Urmatorul element este un div
pozitionat<br> static</p>
<div class="static">
    Acesta este un element
    pozitionat static.
</div>
</body>
```

## Pozitionare statica

Urmatorul element este un div pozitionat static

Acesta este un element  
pozitionat static.

# CSS Layout – proprietatea **position**

## position: relative

Un element cu **position: relative** este poziționat relativ față de poziția pe care ar fi avut-o în mod normal în document;  
Pentru poziționare se folosesc proprietățile left, right, top, bottom.

```
<style>
div{position:relative;
  left: 35px;
  width:200px;
  border: 3px solid #73AD21;
  background-color: cyan;
}
</style>
</head>
<body>
<h2>Poziționare relativa</h2>
<p>Următorul element este un div
poziționat<br>relativ</p>
<div class="relativ">
  Acesta este un element poziționat relativ.
</div>
```

### Poziționare relativa

Următorul element este un div poziționat relativ

Acesta este un element  
poziționat relativ.

# CSS Layout – proprietatea **position**

## position: absolute

Elementul este pozitionat intr-o pozitie relativa fata de primul parinte care are **position:absolute/relative/fixed**; elementul este **scos din fluxul documentului**.

```
<style>
div{position:absolute;
  left: 35px; top:20px;
  width:200px;
  border: 3px solid #73AD21;
  background-color: cyan;
}
</style>
</head>
<body>
<h2>Pozitionare absoluta</h2>
<p>Urmatorul element este un div
pozitionat<br>
absolut</p>
<div class="absolut">
  Acesta este un element pozitionat absolut.
</div>
```

**Po** Acesta este un element **uta**  
pozitionat absolut.

Urmatorul element este un div pozitionat absolut

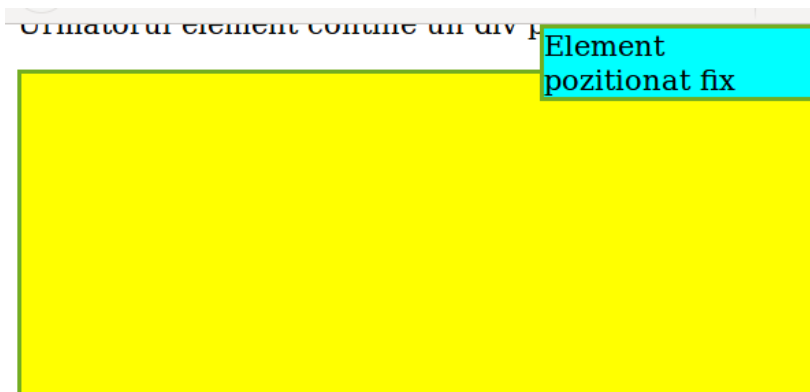
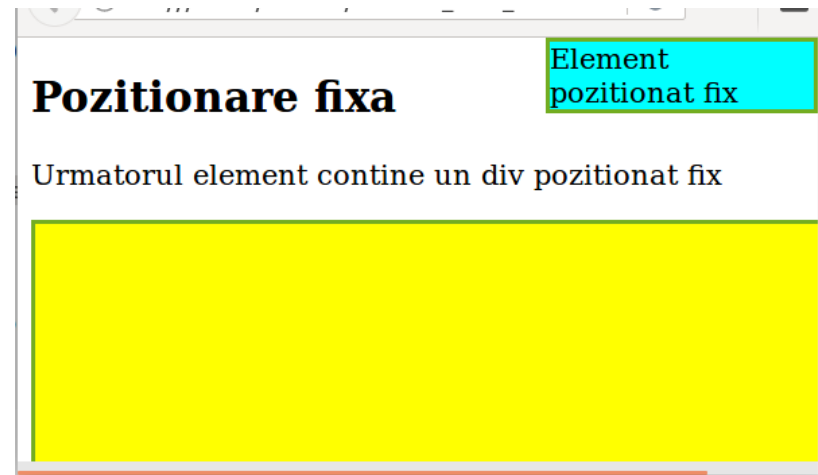
# CSS Layout – proprietatea **position**

## position: fixed

Elementul are o pozitie fixa in fereastra de browser; este scos din fluxul documentului si nu e afectat de scroll.

Este pozitionat folosind proprietatile left, top, right, bottom.

```
<style>
div.parinte{width:500px; height:300px;
  border: 3px solid #73AD21;
  background-color: yellow
}
div.fix{position:fixed;
  right:0; top:0;
  width:150px;
  border: 3px solid #73AD21;
  background-color: cyan;
}
</style>
</head>
<body>
<h2>Pozitionare fixa</h2>
<p>Urmatorul element contine un div pozitionat
fix</p>
<div class="parinte">
<div class="fix">
  Element pozitionat fix
</div>
</div>
```



# CSS Layout – proprietatea **position**

## position: sticky

Elementul comuta între pozitionarea relativa și fixa în funcție de poziția scrolului în interiorul parintelui; dacă poziția scrolului nu depășește elementul acesta va fi pozitionat relativ, altfel se comporta ca un element fix

```
<style>
div.sticky {
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: lightgreen;
  border: 2px solid black;
}
</style>
</head>
<body>
<p>Paragraf 1</p>
<p>Paragraf 2</p>
<p>Paragraf 3</p>
<div class="sticky">Element
sticky</div>
<div>
<p>Paragraf 4</p>
<p>Paragraf 5</p>
<p>Paragraf 6</p>
<p>Paragraf 7</p>
.....
```

Paragraf 1

Paragraf 2

Paragraf 3

Element sticky

Paragraf 4

Paragraf 5

Element sticky

Paragraf 4

Paragraf 5

Paragraf 6

Paragraf 7

Suprapunerea elementelor: proprietatea **z-index**

Prin pozitionare elementele se pot suprapune, se pot crea stive de elemente; ordinea elementelor în stiva este data de **z-index**; Elementele cele mai vizibile au **z-index** mai mare.

Proprietatea **z-index** poate fi setata numai pentru elementele care sunt pozitionate **absolute**, **relative** sau **fixed**.

**z-index: ..., -100 /\* 0, 100, ...**

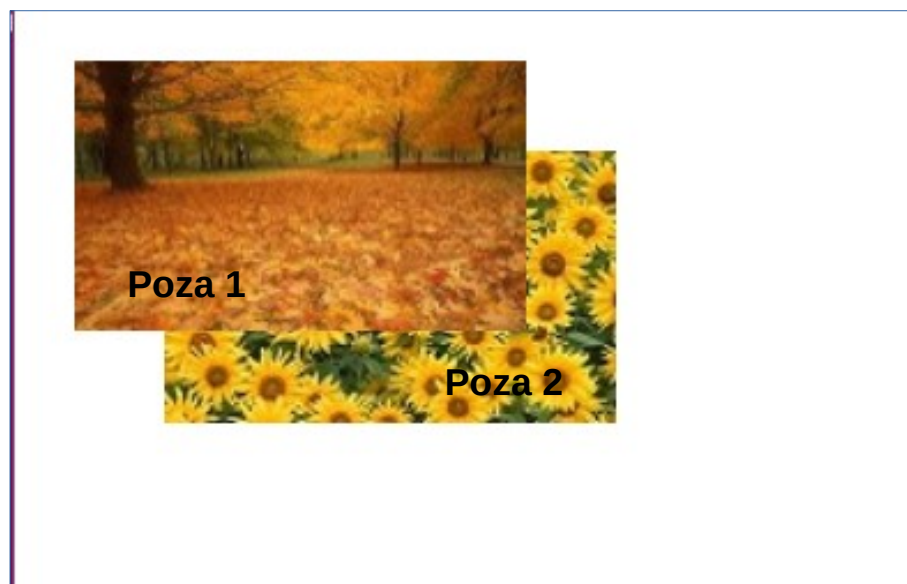


```

<style>
#poza1{position:absolute;
  left:20px; top:20px;
  z-index:10;
}
#poza2{position:absolute;
  left:50px; top:50px;
  z-index:2;
}
</style>
</head>
<body>


</body>

```



```

.....
#poza3{position:absolute;
  left:100px; top:100px;
  z-index:10;
}
</style>
</head>
<body>
.....

</body>

```



CSS layout: proprietatile **float** și **clear**

```
float:left /*right, none*/  
clear:both /*left, right*/
```

Elementele cu proprietatea **float** sunt scoase din fluxul documentului și pozitionate conform valorii, la stanga sau la dreapta. Ele afecteaza celelalte elemente care se aranjeaza inconjurand (“wrapping”) elementele **float**.

Elementele cu proprietatea **clear** nu se aranjeaza in jurul elementelor **float** ci se deplaseaza sub acestea.

**Elementele pozitionate absolut ignora proprietatea float**

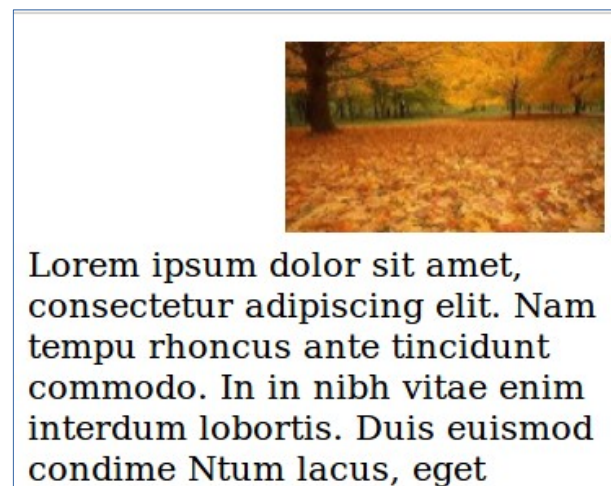
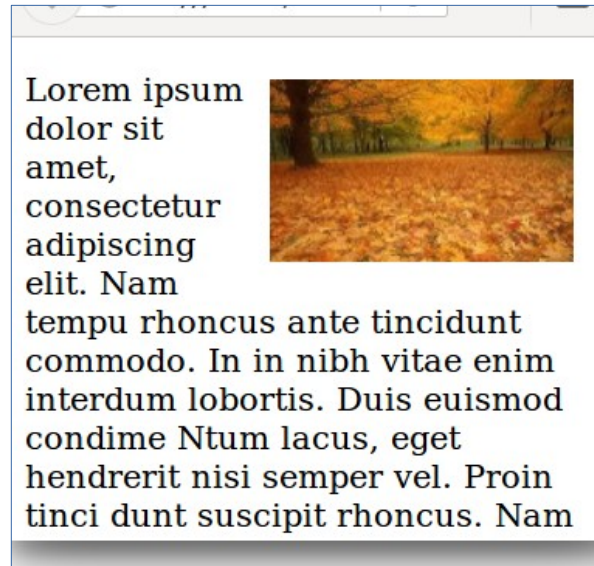
```

<head>
<style>
#poza1 {float:right;
        margin:5px;
}
</style>
</head>
<body>

<p>Lorem ipsum dolor sit
a.....</p>
</body>

```

```
p {clear:both;}
```



# CSS layout - flexbox

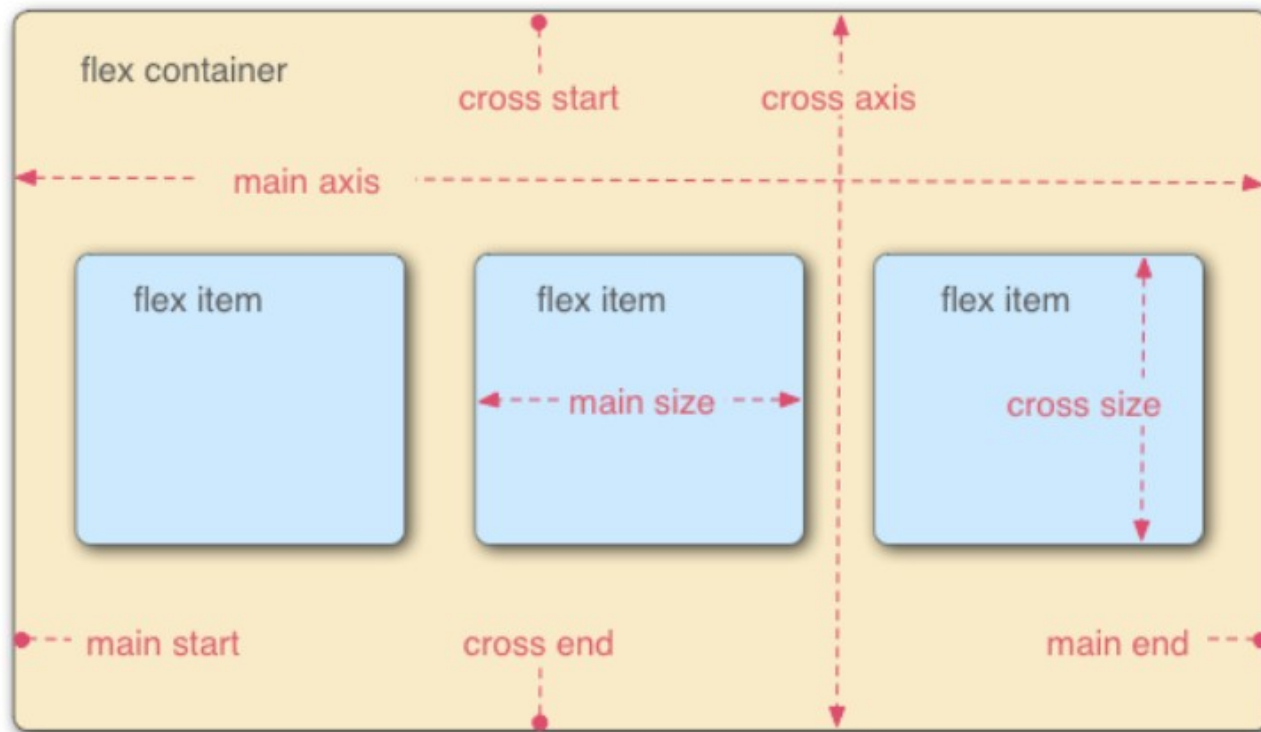
Layoutul flexibil ofera o metoda eficienta de a aseza, alinia și a distribui spațiul între elementele din document chiar și atunci când dimensiunea viewportului și a elementelor este necunoscuta sau dinamica.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Flexbox](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox)

Layoutul flexibil presupune definirea a doua tipuri de elemente:

- Element părinte - **flex container**
- Elemente copii - **flex items**



## Proprietăți pentru părinte (flex container)

display  
flex-direction  
flex-wrap  
flex-flow  
justify-content  
align-items  
align-content

## Proprietăți pentru copii (flex items)

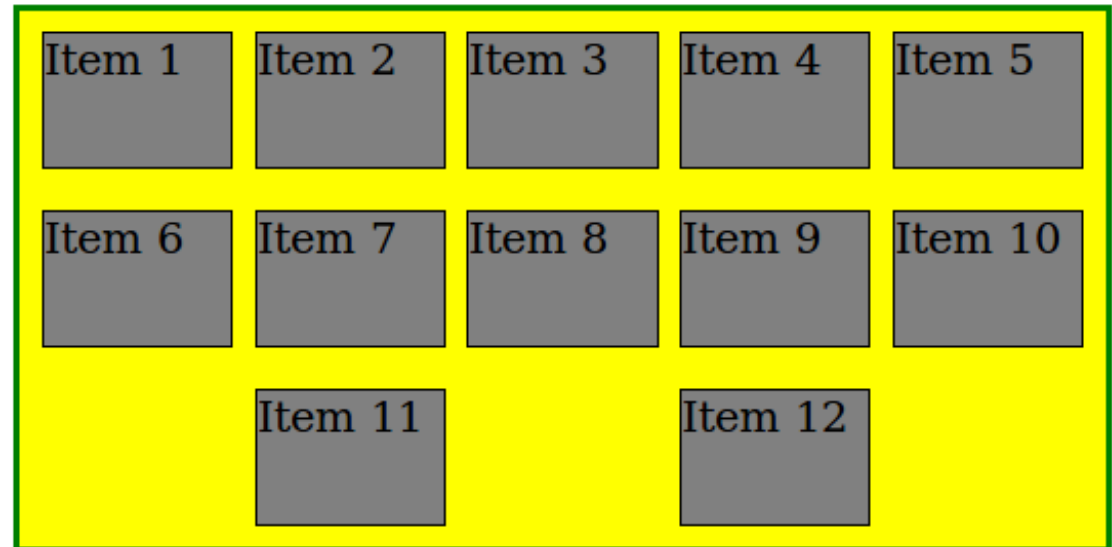
order  
flex-grow  
flex-shrink  
flex-basis  
flex  
align-self

## Exemplu

```
<style>
.container {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-evenly;
    width: 50%; height: 200px;
    border: 3px solid green;
    background-color: yellow;
    padding: 0;
    list-style-type: none;
}

.item {
    align-self: center;
    width: 70px;
    height: 50px;
    border: 1px solid black;
    background-color: grey;
}
</style>
```

```
<body>
<ul class="container">
<li class="item">Item 1</li>
<li class="item">Item 2</li>
<li class="item">Item 3</li>
.....
<li class="item">Item 12</li>
</ul>
</body>
```



# Proprietăți pentru părinte (flex container)

`display: flex /* inline-flex */`

- definește un container flexibil; inline sau bloc în funcție de valoarea dată.
- permite un context flexibil pentru toți copiii săi direcți.

Exemplu:

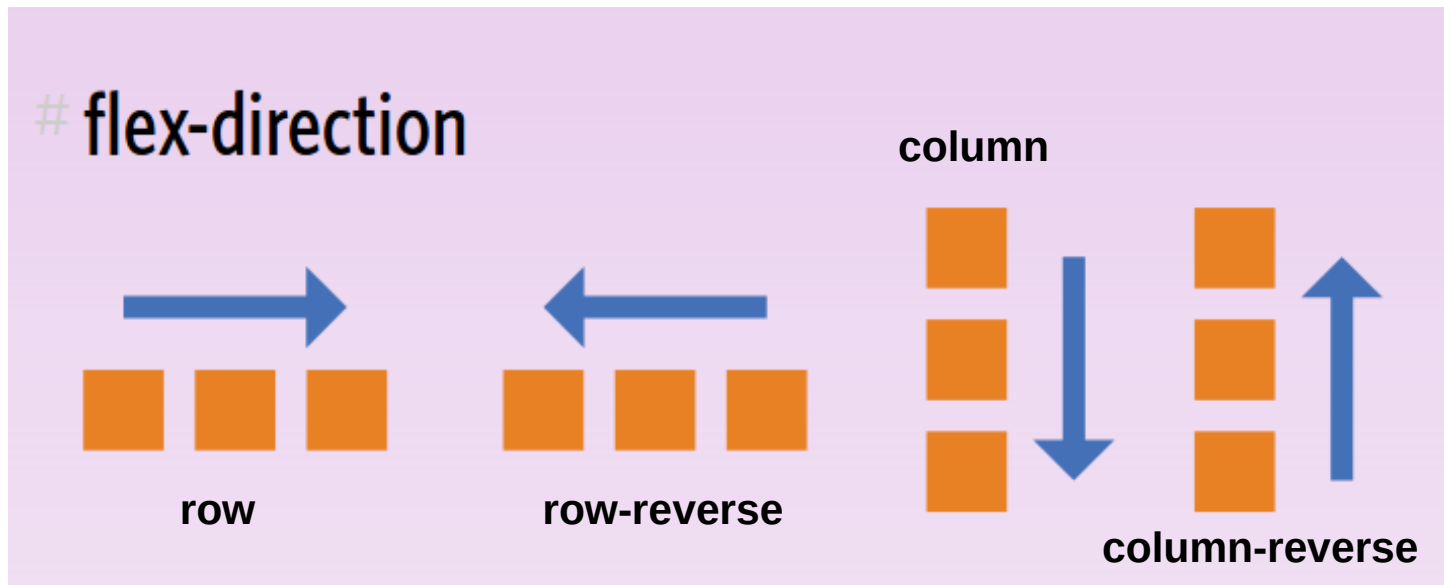
```
ul {display: flex;}  
body {display: flex;}
```



# Proprietăți pentru părinte (flex container)

`flex-direction: row` (implicit) /\* `row-reverse`, `column`, `column-reverse` \*/

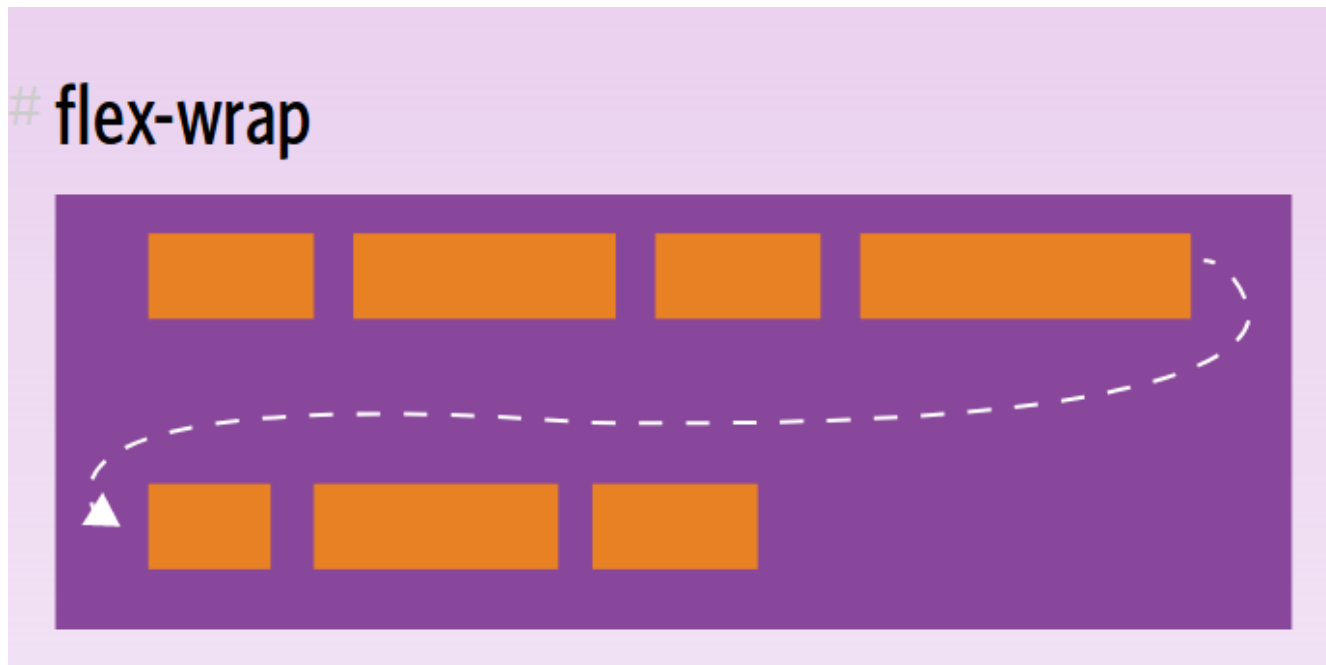
- stabilește direcția în care sunt plasate itemurile în container: pe orizontala (stânga-dreapta sau dreapta-stanga) sau pe verticala (sus-jos sau jos-sus).



# Proprietăți pentru părinte (flex container)

**flex-wrap: wrap** /\* nowrap (implicit) , wrap-reverse \*/

- stabilește dacă itemurile vor fi aranjate pe o singură linie (coloana) sau pe mai multe; implicit ele se vor aranja toate pe o linie (coloana).

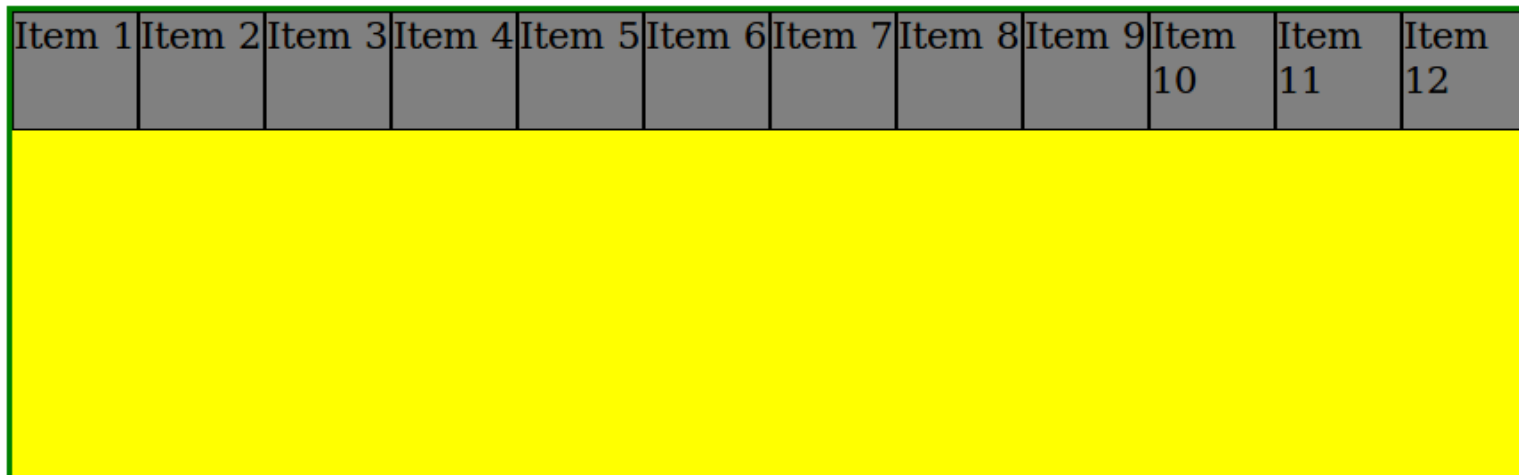


# Proprietăți pentru părinte (flex container)

**flex-flow: flex-direction flex-wrap**

- prescurtare pentru proprietatile flex-direction și flex-wrap;
- implicit: **flex-flow: row nowrap**

```
.container {  
  display: flex;  
  flex: row nowrap;  
}
```



# Proprietăți pentru părinte (flex container)

**justify-content: flex-start** (implicit)  
/\* flex-end,  
center,  
space-between,  
space-around,  
space-evenly \*/

- specifica modul cum sunt aliniate itemurile de-a lungul axei principale

## # justify-content

**flex-start**



**flex-end**



**center**



**space-between**



**space-around**



**space-evenly**

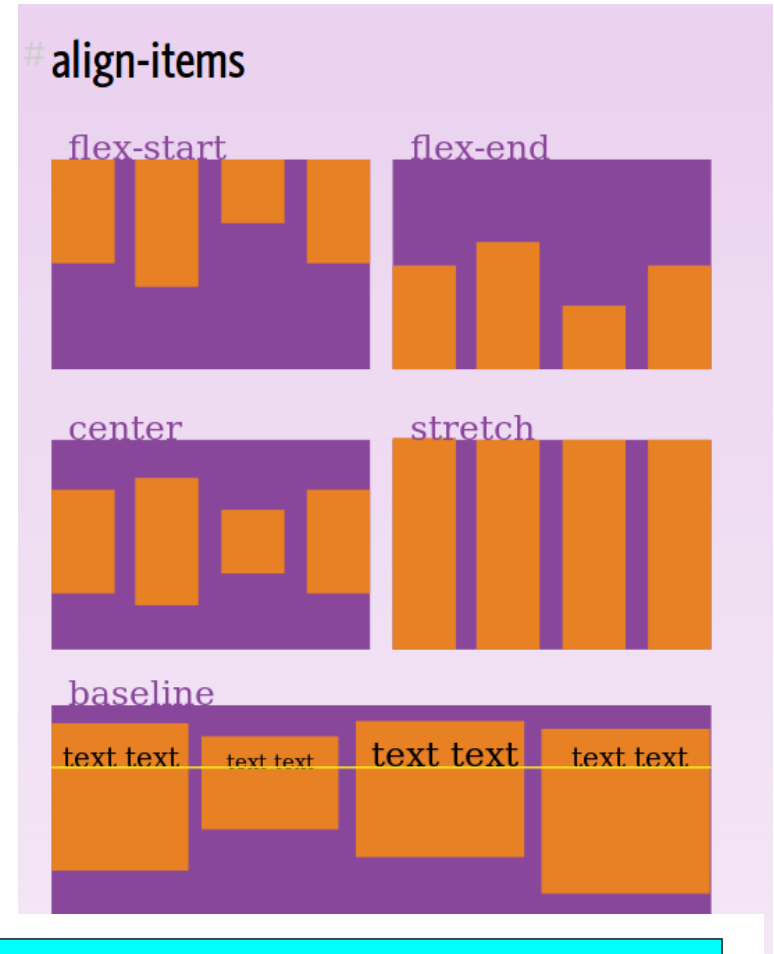


# Proprietăți pentru părinte (flex container)

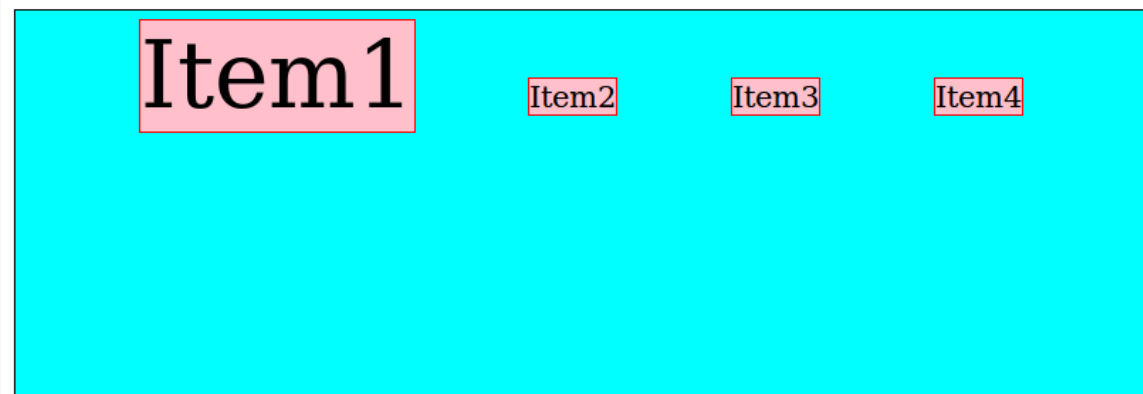
`align-items: flex-start`

`/* flex-end,  
center,  
stretch (implicit),  
baseline */`

- specifica modul cum sunt aliniate itemurile pe axa secundara (axa perpendiculara pe axa principala)



`align-items: baseline`



# Proprietăți pentru părinte (flex container)

`align-content: flex-start`

`/* flex-end,  
center,  
stretch (implicit),  
space-between,  
space-around */`

- aliniază liniile containerului dacă există un spațiu suplimentar pe axa transversală;
- nu are efect dacă există o singură linie în container.

# align-content

flex-start



flex-end



center



stretch



space-between



space-around



## Proprietăți pentru copii (flex items)

`order` || `flex-grow` || `flex-shrink` || `flex-basis` || `flex` || `align-self`

`order: <intreg>` (implicit este 0)

- stabilește ordinea în care itemurile apar în containerul flexibil;
- implicit ele apar în ordinea în care sunt scrise în documentul HTML.

```

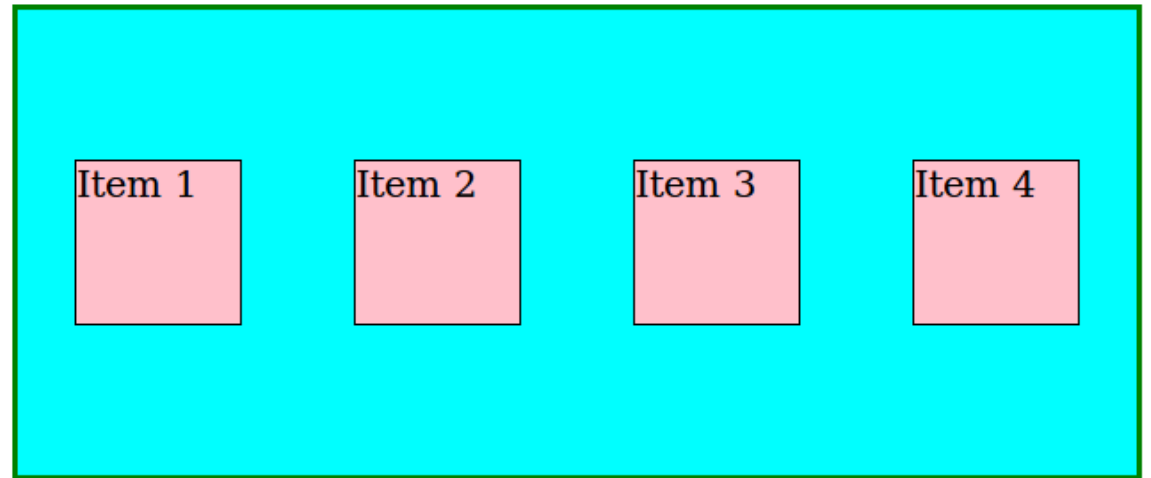
ul {
  display : flex;
  flex-flow: row nowrap;
  justify-content:space-around;
  align-items: center;
  width:50%;
  height:200px;
  border:3px solid green;
  background-color:cyan;
  padding:0;
  list-style-type:none;
}
li {
  width:70px;
  height:70px;
  border:1px solid black;
  background-color:pink;
}

```

```

<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ul>

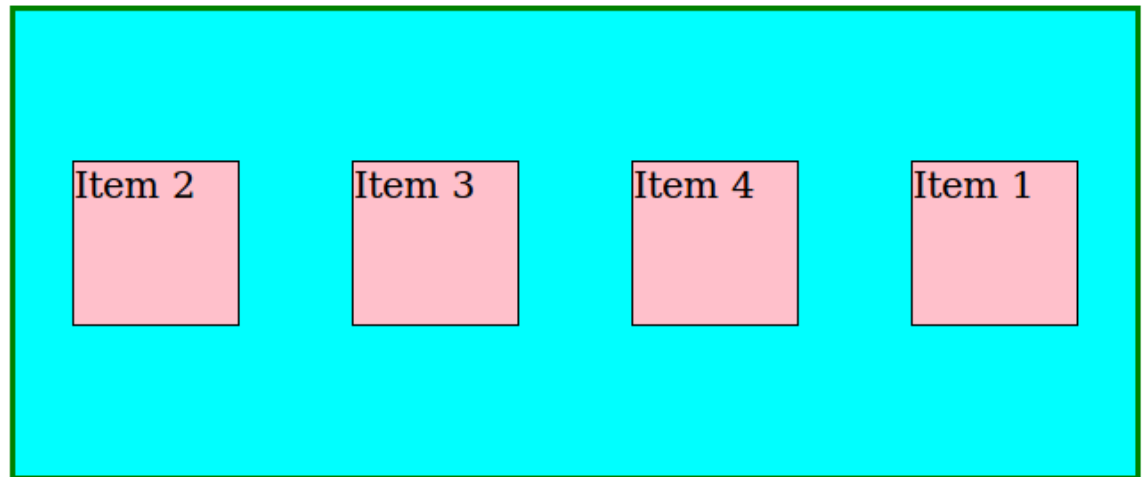
```



Ex.1

Ex.1

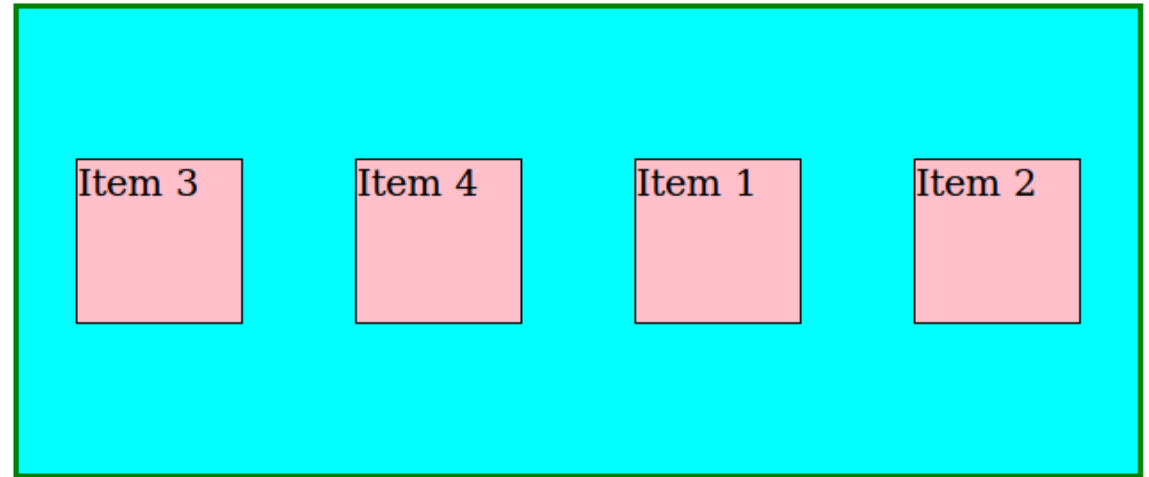
```
li:nth-child(1) { order:1; }
```





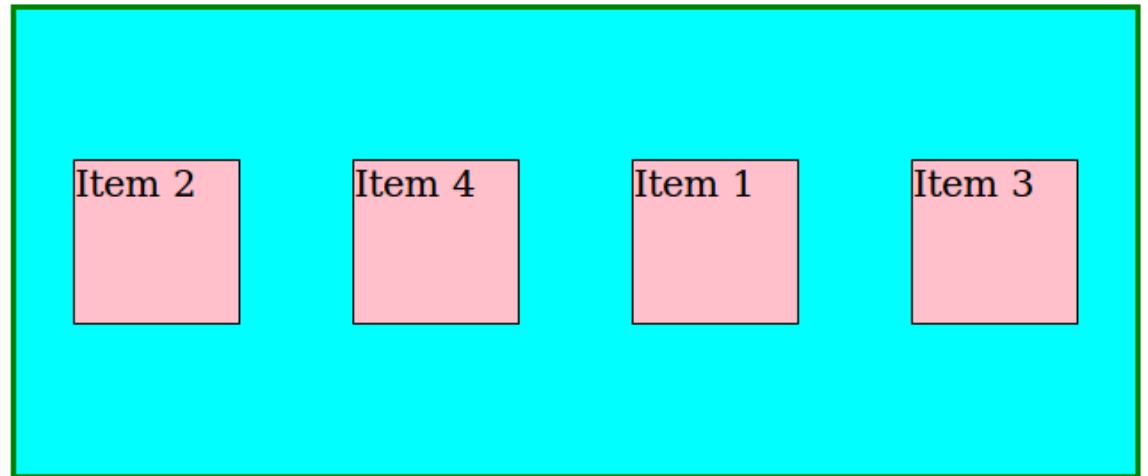
Ex.2

```
li:nth-child(1) {order:1;}  
li:nth-child(2) { order:2;}
```



Ex.3

```
li:nth-child(1) {order:1;}  
li:nth-child(3) { order:1;}
```



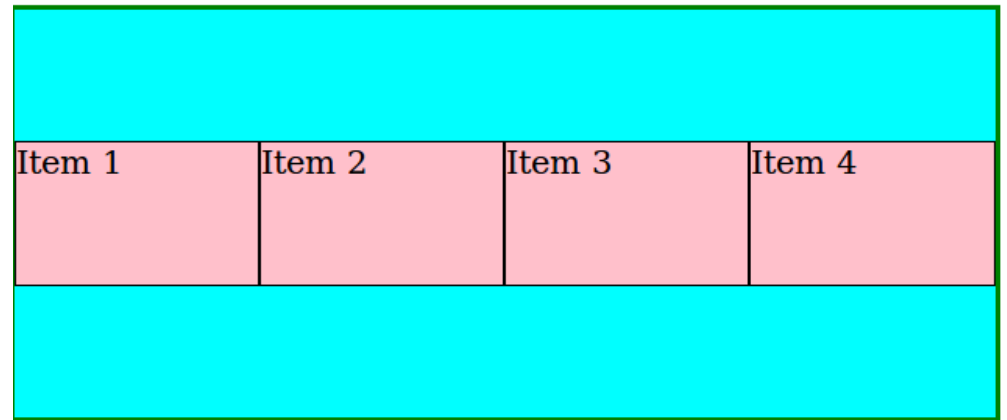
## Proprietăți pentru copii (flex items)

order || **flex-grow** || flex-shrink || flex-basis || flex || align-self

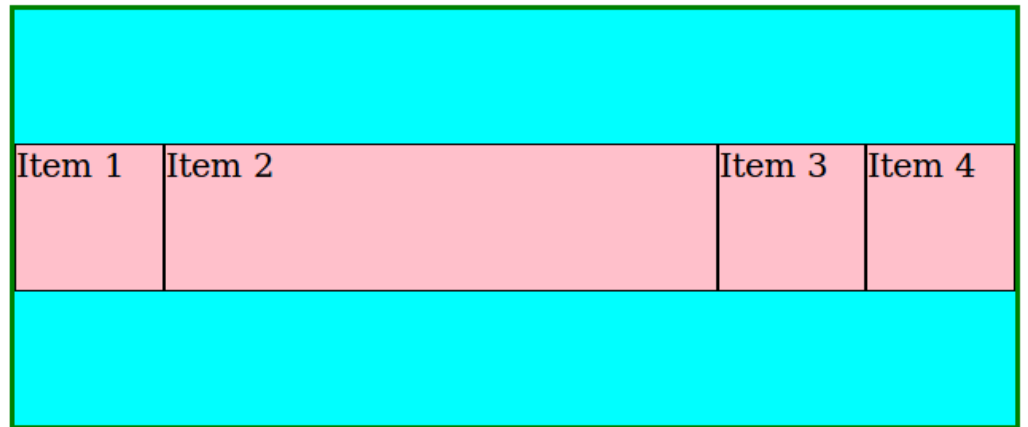
**flex-grow: <number>** (implicit este 0)

- stabilește capacitatea itemurilor de a-si mari dimensiunea dacă este spațiu disponibil în interiorul containerului;
- dacă toate itemurile au flex-grow egal cu 1, ele vor împărți spațiul disponibil în mod egal.

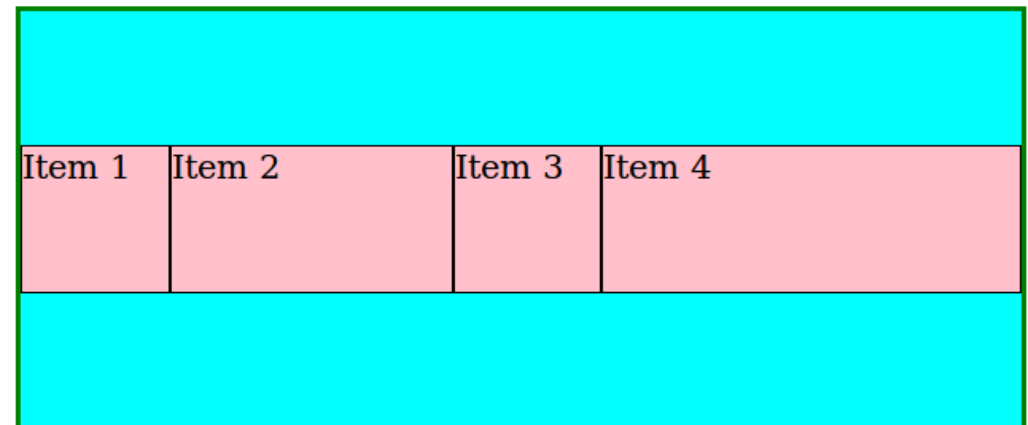
```
li { flex-grow:1; }
```



```
li:nth-child(2) { flex-grow:1; }
```



```
li:nth-child(2) { flex-grow:1; }  
li:nth-child(4) { flex-grow:2; }
```



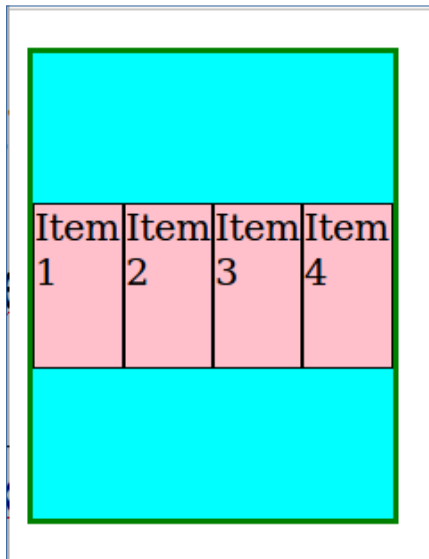
# Proprietăți pentru copii (flex items)

order || flex-grow || **flex-shrink** || flex-basis || flex || align-self

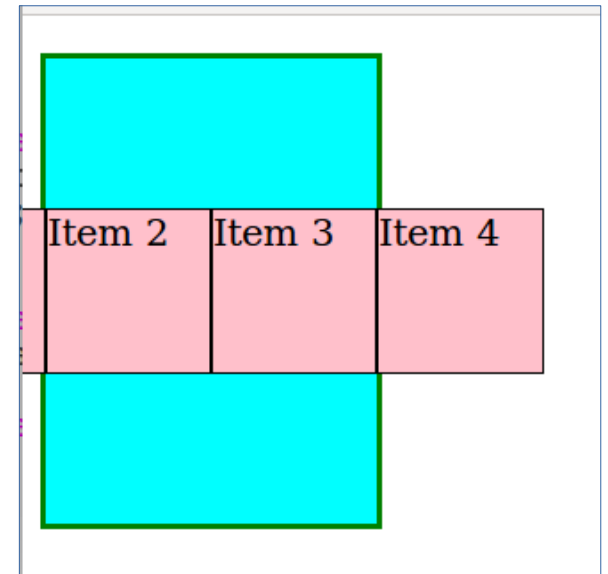
**flex-shrink:** <number> (implicit este 1)

- definește capacitatea itemurilor de a se micșora dacă este necesar

Implicit (flex-shrink:1)



li {flex-shrink:0;}

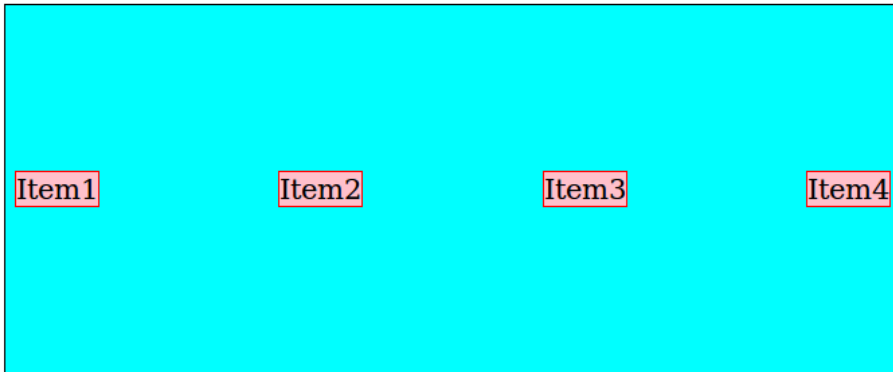


## Proprietăți pentru copii (flex items)

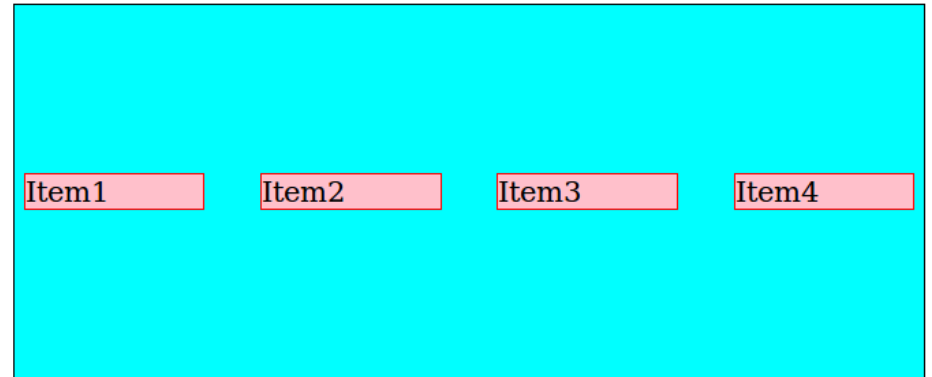
order || flex-grow || flex-shrink || **flex-basis** || flex || align-self

**flex-basis:** <length> /\* auto (implicit)\*/

- definește dimensiunea implicită a unui element înainte de distribuirea spațiului rămas;



flex-basis: auto



flex-basis: 100px

## Proprietăți pentru copii (flex items)

order || flex-grow || flex-shrink || flex-basis || **flex** || align-self

**flex: flex-grow flex-shrink flex-basis**

- prescurtare pentru proprietatile flex-grow, flex-shrink si flex-basis;
- Implicit este **flex: 0 1 auto**

```

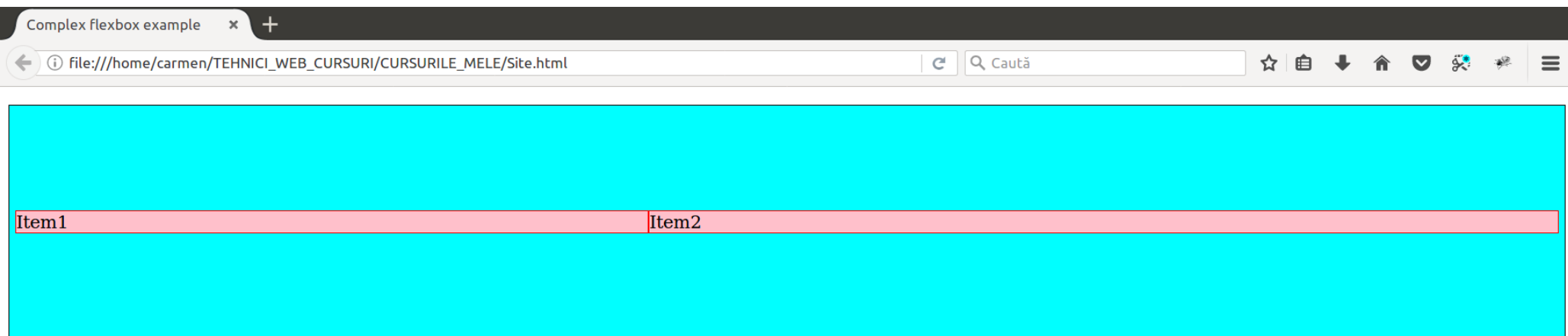
<style>
ul { display:flex;
      flex-flow: row nowrap;
      align-items: center;
      justify-content:space-between;
      border:1px solid black;
      padding:5px;
      background-color:cyan;
      height:200px;
      list-style-type:none;
    }
li { background-color:pink;
     border:1px solid red;
   }
li:nth-child(1)
  { flex:1 1 20em;}
li:nth-child(2)
  { flex:2 2 20em;}
</style>

```

```

<body>
<ul>
<li>Item1</li>
<li>Item2</li>
</ul>
</body>

```



## Proprietăți pentru copii (flex items)

order || flex-grow || flex-shrink || flex-basis || flex || align-self

align-self: flex-start

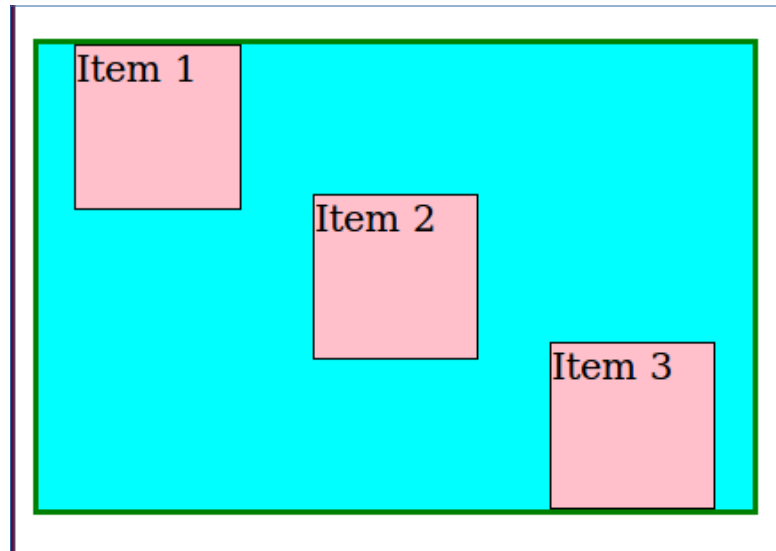
/\* flex-end,  
center,  
stretch,  
baseline,  
auto\*/

- Permite alinierea individuala a itemurilor pe axa secundara (inlocuieste alinierea specificata cu align-items)



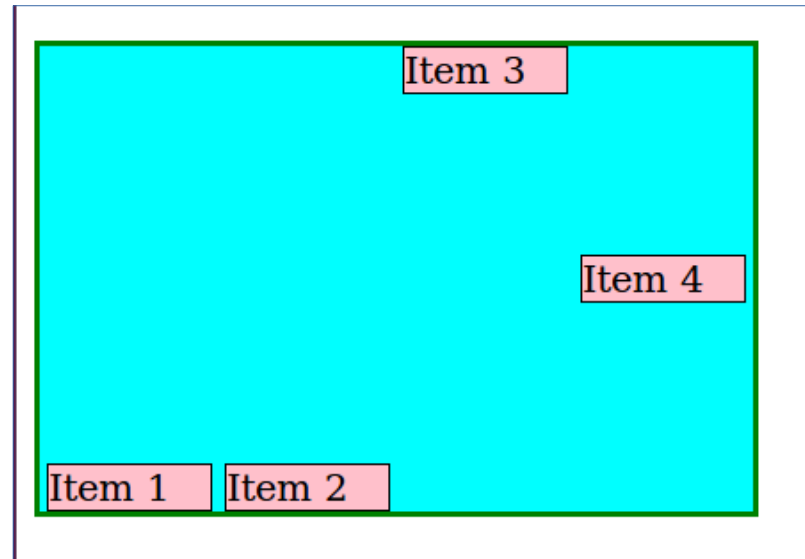
```
ul{display:flex;  
  flex-flow: row wrap;  
  justify-content:space-around;  
  align-items:center; }
```

```
li:nth-child(1){align-self: flex-start;}  
li:nth-child(3){align-self: flex-end;}
```



```
ul {display:flex;  
  flex-flow: row wrap;  
  justify-content:space-around;  
  align-items:flex-end;}
```

```
li:nth-child(1){align-self: auto;}  
li:nth-child(3){align-self: baseline;}  
li:nth-child(4){align-self: center;}
```



## Galerie de imagini folosind display:flex

```
<section id="container">
<figure class="item">

<figcaption> flori de primavara
</figcaption>

</figure>
<figure class="item">

<figcaption> peisaj de toamna
</figcaption>
</figure>
```

.....

```
#container {display: flex;
            flex-flow: row wrap;
            width:70%;
            border:1px solid black;
            }
```

```
.item { flex: 1 0 auto;
        margin: 1em;}
```

```
.imgf {width: 100%; height:auto;}
```



flori de primavara



peisaj de toamna



flori de vara



peisaj de iarna



flori de primavara



peisaj de toamna



flori de vara



flori de primavara



peisaj de toamna



flori de vara



peisaj de iarna



flori de primavara

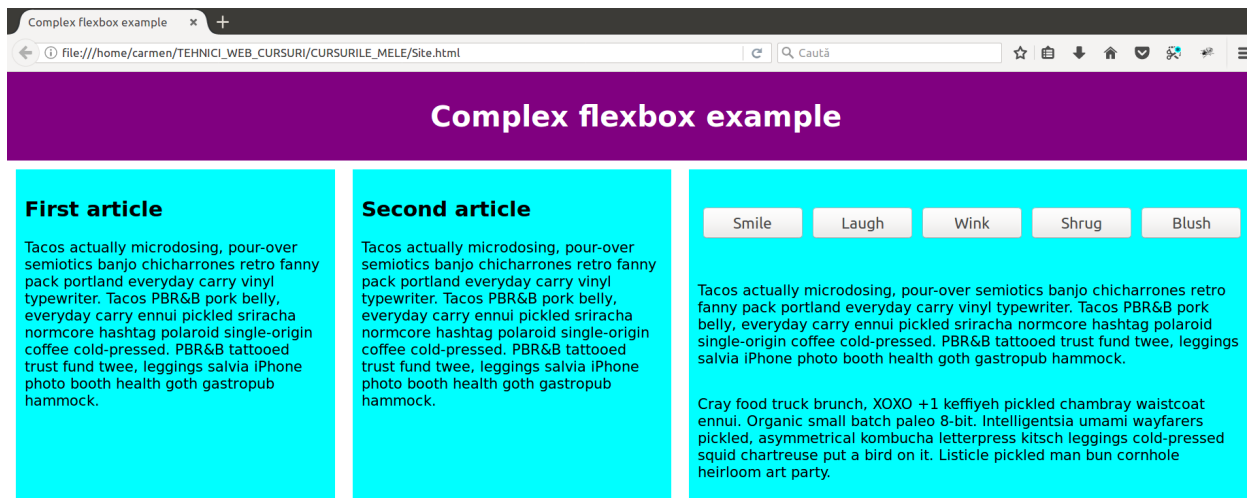


peisaj de toamna

```

section - article
      article
    article - div - button
                div    button
                div    button
                button
                button

```



```

section {
  display: flex;
}
article {
  flex: 1 200px;
}
article:nth-of-type(3) {
  flex: 3 200px;
  display: flex;
  flex-flow: column;
}
article:nth-of-type(3) div:first-
child {
  flex: 1 100px;
  display: flex;
  flex-flow: row wrap;
  align-items: center;
  justify-content: space-around;
}
button {
  flex: 1;
  margin: 5px;
  font-size: 18px;
  line-height: 1.5;
}

```

# CSS Grid

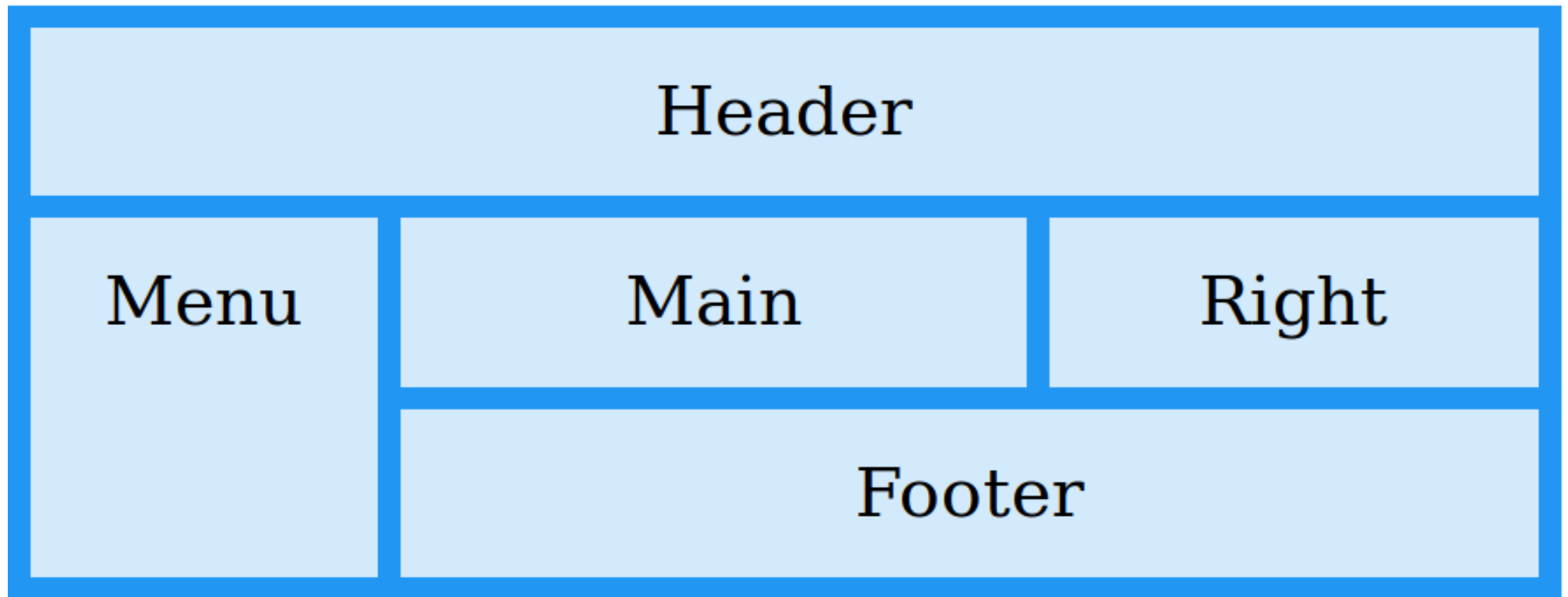
CSS Grid ofera un sistem de layout bazat pe rețea, cu rânduri și coloane, usurand astfel modul de a aseza, alinia și a distribui spațiul între elementele din document.

<https://css-tricks.com/snippets/css/complete-guide-grid/>

[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Grid\\_Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout)

CSS Grid presupune definirea a doua tipuri de elemente:

- Element părinte – **grid container**
- Elemente copii - **grid items**



CSS Grid format din 5 itemuri aranjate pe 6 coloane și 3 randuri

# CSS Grid

## Proprietăți pentru părinte (grid container)

display  
grid-template-columns  
grid-template-rows  
grid-column-gap  
grid-row-gap  
justify-content  
align-content  
justify-items  
align-items  
grid-template-areas

## Proprietăți pentru copii (grid items)

grid-column  
grid-row  
grid-area  
justify-self  
align-self  
place-self

## Exemplu

### CSS

```
.grid-container {  
  display: grid;  
  grid-template-columns: auto auto auto;  
  background-color: yellow;  
  padding: 10px;  
}  
.grid-item {  
  background-color: grey;  
  border: 1px solid rgba(0, 0, 0, 0.8);  
  padding: 20px;  
  font-size: 30px;  
  text-align: center;  
}
```

### HTML

```
<body>  
<div class="grid-container">  
  <div class="grid-item">1</div>  
  <div class="grid-item">2</div>  
  <div class="grid-item">3</div>  
  <div class="grid-item">4</div>  
  <div class="grid-item">5</div>  
  <div class="grid-item">6</div>  
  <div class="grid-item">7</div>  
  <div class="grid-item">8</div>  
  <div class="grid-item">9</div>  
</div>  
</body>
```

1	2	3
4	5	6
7	8	9

# Proprietăți pentru părinte (grid container)

`display: grid /* inline-grid*/`

- definește elementul ca un grid container; inline sau bloc în funcție de valoarea dată.
- stabilește un context de formatare a grilei pentru toți copiii săi direcți.

Exemplu:

```
div {display: inline-grid;}  
body {display: grid;}
```

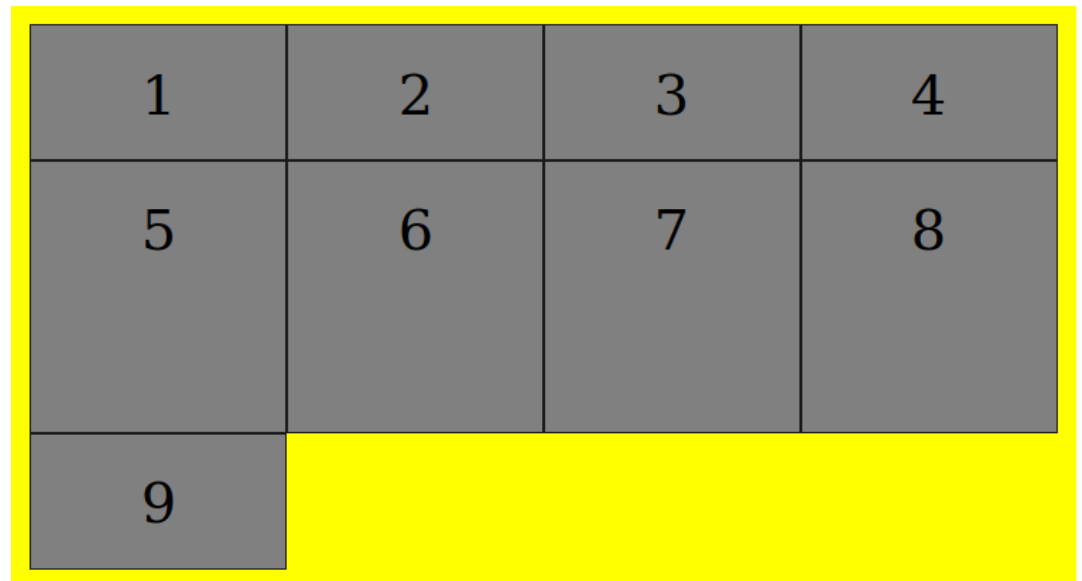


# Proprietăți pentru părinte (grid container)

`grid-template-columns`  
`grid-template-rows`

- definește numărul de coloane și rânduri în care este împărțit gridul și lățimea/înălțimea acestora (unități de măsură: %, px, auto, fr)
- funcții: **repeat**(count,size), count: număr, auto-fill, auto-fit  
**minmax**(min,max)

```
.grid-container {  
  display: grid;  
  grid-template-columns: auto auto auto auto;  
  grid-template-rows: 25% auto 25%;  
  background-color: yellow;  
  padding: 10px;  
  height: 300px;  
}
```

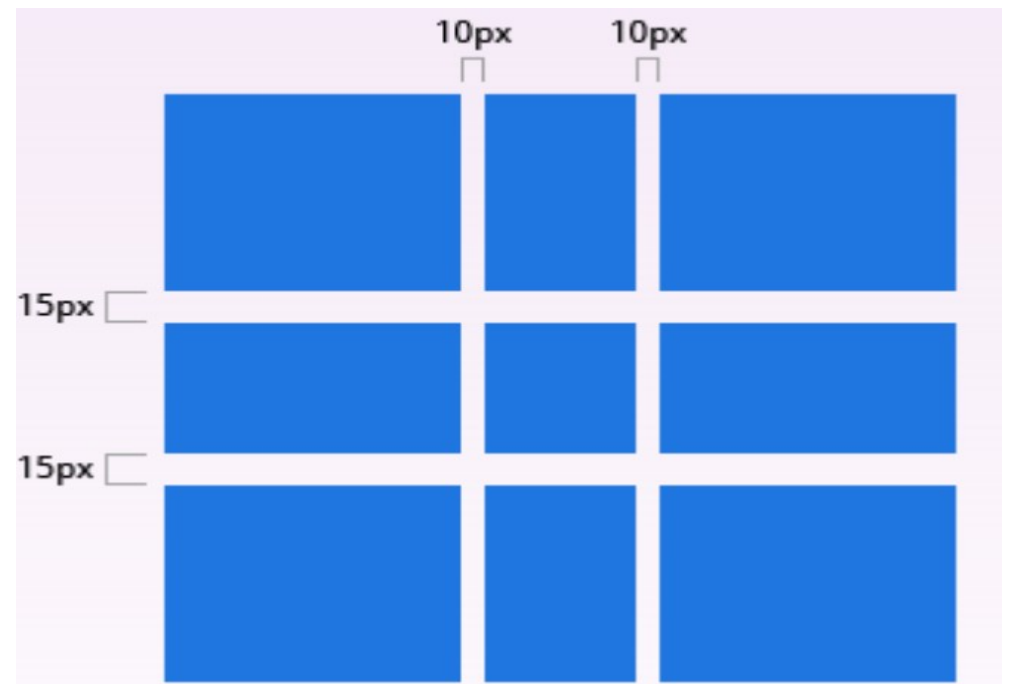


# Proprietăți pentru părinte (grid container)

`grid-column-gap`  
`grid-row-gap`

- definește lățimea liniilor rețelei (spațiul dintre coloane și dintre randuri);
- `grid-gap`: prescurtare pentru proprietățile `grid-row-gap` și `grid-column-gap`;

```
.grid-container {  
  grid-template-columns: 100px 50px  
  100px;  
  grid-template-rows: 80px auto 80px;  
  grid-column-gap: 10px;  
  grid-row-gap: 15px;  
}
```



# Proprietăți pentru părinte (grid container)

`justify-content: start` (implicit)

`/* end,`

`center,`

`stretch,`

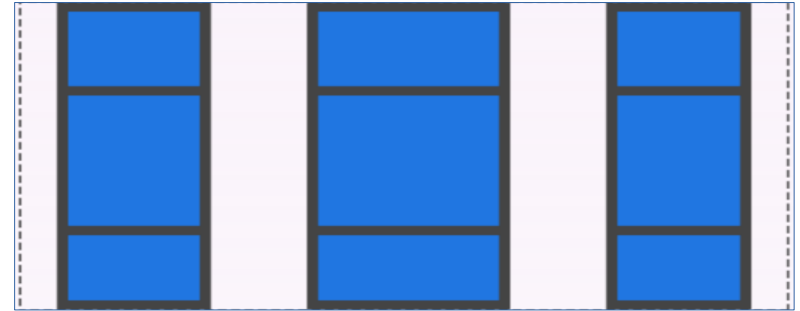
`space-around,`

`space-between,`

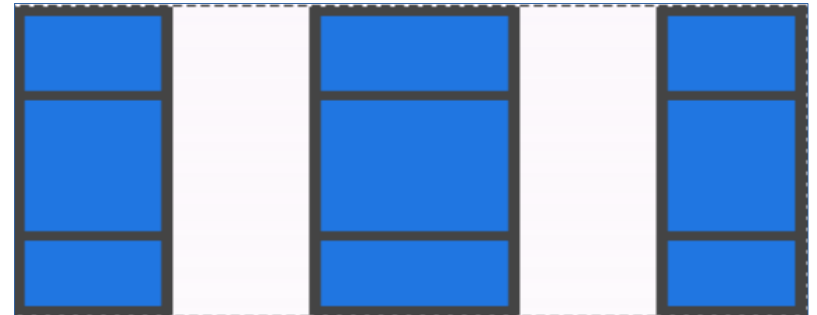
`space-evenly */`

- specifica modul cum este aliniata întreaga grila în interiorul containerului
- alinierea se face pe orizontala

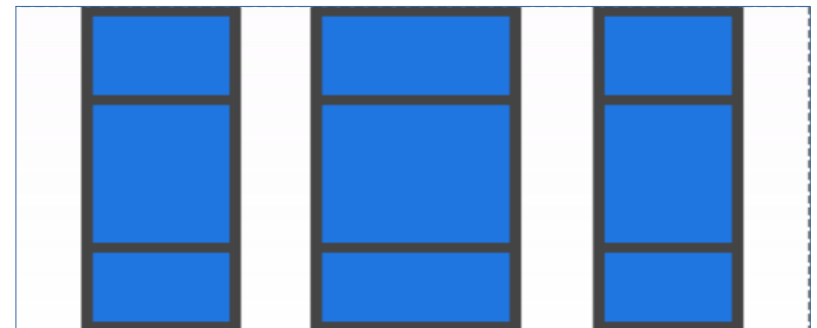
space-around



space-between



space-evenly



# Proprietăți pentru părinte (grid container)

`align-content: start` (implicit)

`/* end,`

`center,`

`stretch,`

`space-around,`

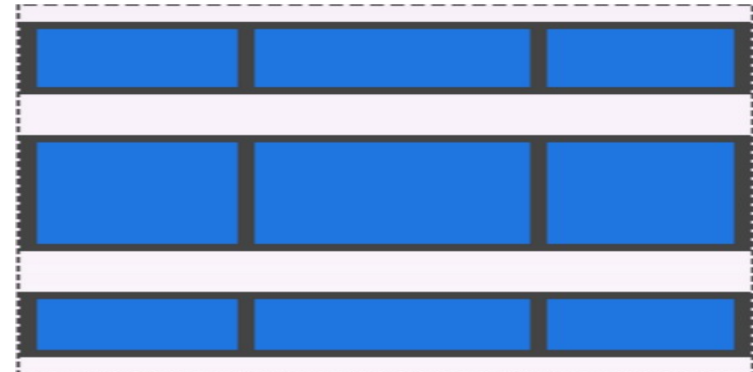
`space-between,`

`space-evenly */`

- specifica modul cum este aliniata grila în interiorul containerului, pe verticala

`place-content: align-content justify-content`

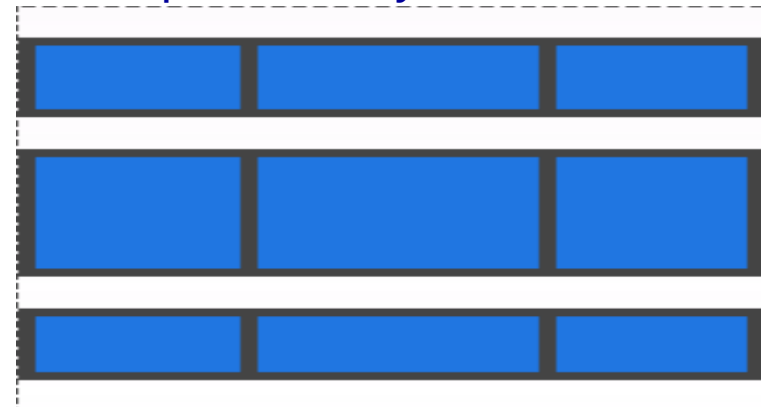
space-around



space-between



space-evenly



# Proprietăți pentru părinte (grid container)

**justify-items:** start

/\* end,  
center,  
stretch (implicit)

- specifica modul cum sunt aliniate itemurile pe axa principala

**align-items:** start

/\* end,  
center,  
stretch (implicit)

- specifica modul cum sunt aliniate itemurile pe axa secundara (axa perpendiculara pe axa principala)

**place-items:** align-items justify-items

## Proprietăți pentru copii (grid items)

**grid-column: start-coloana / end-coloana**

- definește coloanele în care se plasează un element
- este o prescurtare pentru proprietatile **grid-column-start** și **grid-column-end**

**grid-row: start-linie / end-linie**

- definește randurile în care se plasează un element
- este o prescurtare pentru proprietatile **grid-row-start** și **grid-row-end**

## Exemple

```
.item1 {  
  grid-column: 1 / 5;  
  
  /* grid-column: 1/ span 4; */  
}
```

1				2	3
4	5	6	7	8	9
10	11	12	13	14	15

```
.item1 {  
  grid-row: 1 / 4;  
  
  /* grid-row: 1 / span 3; */  
}
```

1	2	3	4	5	6
	7	8	9	10	11
	12	13	14	15	16

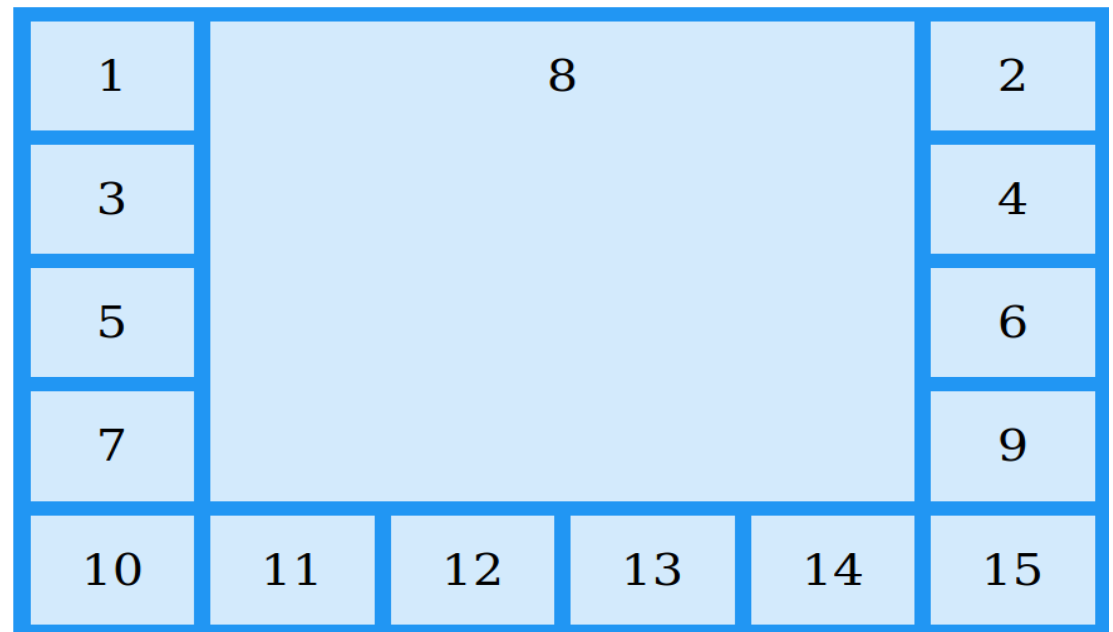
## Proprietăți pentru copii (grid items)

grid-area: start-linie / start-coloana / end-linie / end-coloana

este o prescurtare pentru proprietatile

grid-row-start, grid-column-start, grid-row-end, grid-column-end

```
.item8 {  
  grid-area: 1 / 2 / 5 / 6;  
}
```

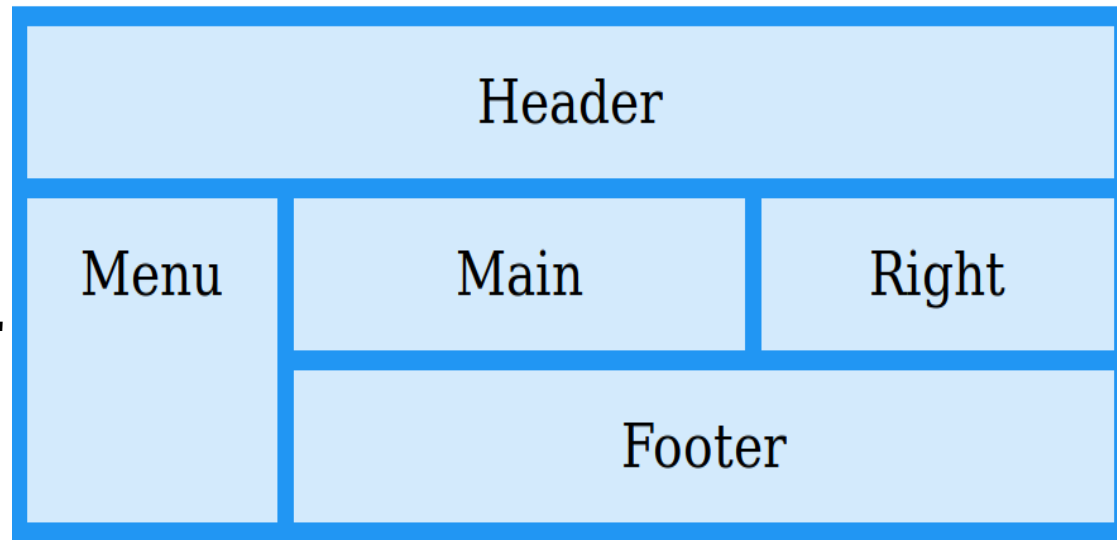




## grid-area: nume-item

poate denumi un item al gridului care este referit apoi folosind proprietatea **grid-template-areas** pentru container

```
.item1 { grid-area: header; }  
.item2 { grid-area: menu; }  
.item3 { grid-area: main; }  
.item4 { grid-area: right; }  
.item5 { grid-area: footer; }  
  
.grid-container {  
  display: grid;  
  grid-template-areas:  
    'header header header header header header'  
    'menu main main main right right'  
    'menu footer footer footer footer footer';  
  grid-gap: 10px;  
  background-color: #2196F3;  
  padding: 10px;  
}
```



# Proprietăți pentru copii (grid items)

**justify-self: start**

**/\* end,  
center,  
stretch** (implicit)

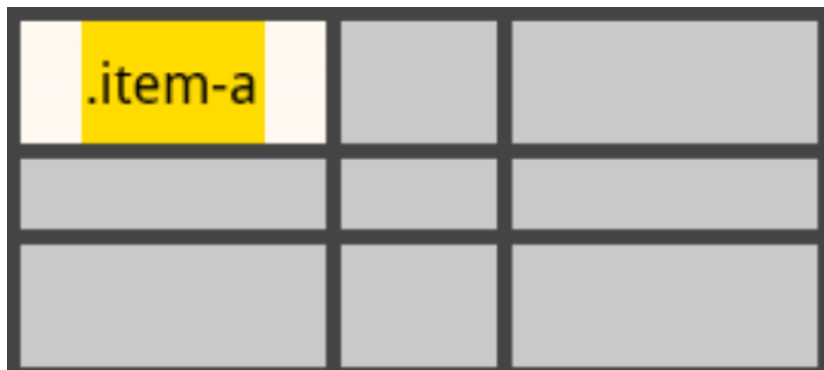
- aliniaza un item al gridului în interiorul celulei, pe axa principala

**align-self: start**

**/\* end,  
center,  
stretch** (implicit)

- aliniaza un item al gridului în interiorul celulei, pe axa secundara

```
.item-a {  
  justify-self: center;  
}
```



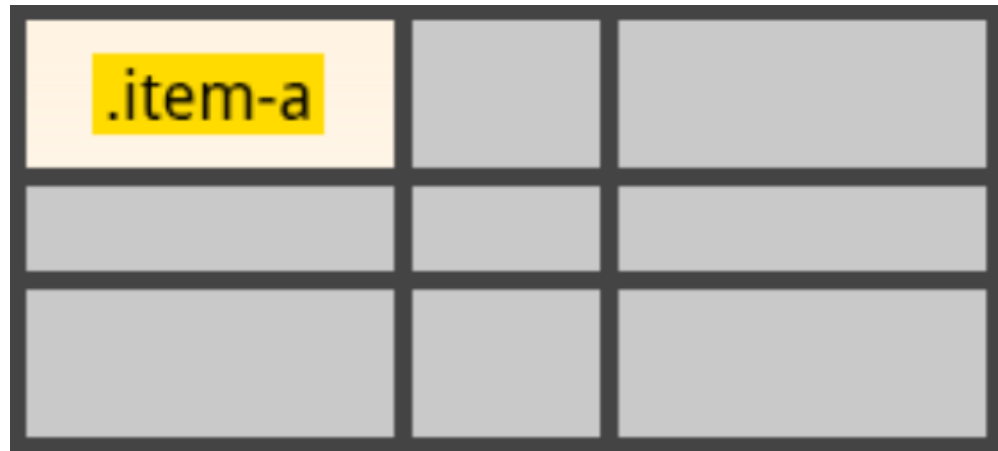
```
.item-a {  
  align-self: center;  
}
```



# Proprietăți pentru copii (grid items)

**place-self:** align-self justify-self

```
.item-a {  
  place-self: center;  
}
```



## Exercitiu

**Scrieți cod CSS astfel încât divurile din documentul HTML să fie aranjate ca în figura de mai jos**

1	2	3
		4
		5

## HTML

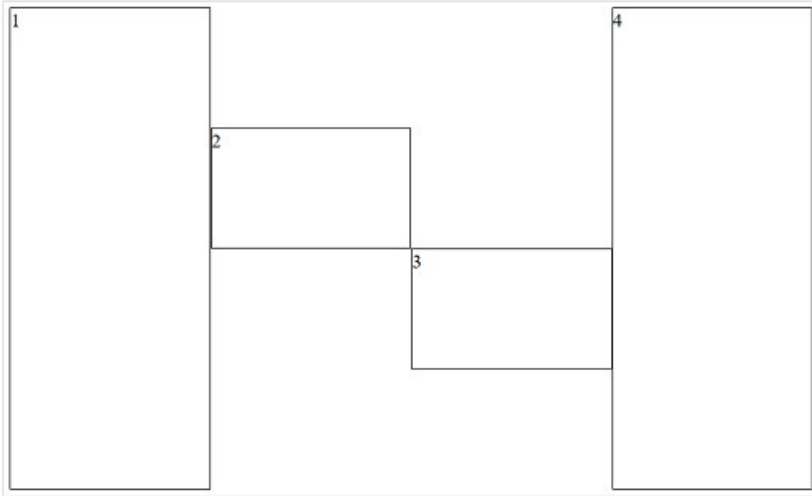
```
<div id="container">  
    <div id="d1">1</div>  
    <div id="d2">2</div>  
    <div id="d3">3</div>  
    <div id="d4">4</div>  
    <div id="d5">5</div>  
</div>
```

## CSS -grid

```
#container{  
    display:grid;  
    grid-template-columns: 1fr 1fr 1fr;  
    grid-template-rows: 1fr 1fr 1fr;  
}  
#d1, #d2{  
    grid-row: 1 / 4; }
```

## Exercitiu

**Scrieți cod CSS astfel încât divurile din documentul HTML să fie aranjate ca în figura de mai jos**



### CSS -grid

```
#container{  
  display:grid;  
  grid-template-columns: 1fr 1fr 1fr 1fr ;  
  grid-template-rows: 100px 100px 100px 100px;  
}
```

### HTML

```
<div id="container">  
  <div id="d1">1</div>  
  <div id="d2">2</div>  
  <div id="d3">3</div>  
  <div id="d4">4</div>  
</div>
```

```
#d1{ grid-column:1/2;  
      grid-row: 1/5;  
}
```

```
#d4{ grid-column: 4/5;  
      grid-row: 1/5;  
}
```

```
#d2{ grid-column: 2/3;  
      grid-row: 2/3;  
}  
#d3{  
  grid-column: 3/4;  
  grid-row: 3/4;  
}
```